POCKET COMPUTER NEWSLETTER

© Copyright 1983 POCKET COMPUTER NEWSLETTER

Issue 30 - December

CARLETON 16K EPROM MODULES

Carleton Micro Systems has announced the availability of a 16K EPROM Module for the Sharp PC-1500 and PC-1500A or the Radio Shack PC-2 computers. Using this new module, program source codes can be stored permanently. Furthermore, Carleton states that the source code in these modules can be protected against any kind of viewing, copying or editing. This is accomplished by disabling the LIST, LLIST, CSAVE and editing functions.

The firm claims that these modules are much more reliable than battery backup RAM modules. Using these modules, complete software packages may be quickly interchanged by simply plugging them into the computer hardware.

Using a Carleton EPROM Module with a Sharp PC-1500A results in a system containing 16K of program ROM and 8K of data RAM.

The company can either burn a program into the module or assist software vendors in programming the modules on their own. Normally, software

Another *Personal Information*™ product.

received on cassette or battery backup RAM is transferred to the Carleton ROM Modules. Carleton works under standard non-disclosure terms to protect vendors software. Orders for as many as 250 ROM Modules can be processed in less than a week.

The service is primarily designed to support software vendors who wish to produce modules in quantity. A minimum module order is normally 25 pieces. At this quantity, a 16K EPROM Module is priced at $150.00 each, plus an initial programming setup fee. The price declines to the range of $100.00 per module for higher quantities.

For further information contact: **Carleton Micro Systems,** *1801 Commerce Drive, South Bend, IN 46624.* The phone number is (219) 236-4630.

HEWLETT-PACKARD ANNOUNCES HP-41CX

Hewlett-Packard Corporation keeps striving to bring pocket computer capabilities to its line of Series 40 advanced programmable calculators. Now the firm has added clock and calendar functions, plus text-file editing and extended memory to its HP-41 line in its new HP-41CX.

The new model has all the features of its predecessor, the HP-41CV, plus those mentioned in the preceding paragraph, 20 new commands, and over 3,100 bytes of memory.

The Time Module in the HP-41CX enables the operator to use the highly portable unit as a time-based controller, an alarm clock, an appointment reminder, a timer or a stopwatch.

The Extended Functions/Memory Module in the HP-41CX (which was an option in the 41CV) provides 868 bytes of extended memory, memory-management functions, programmable versions of several HP-41 functions, plus several register and flag manipulation functions.

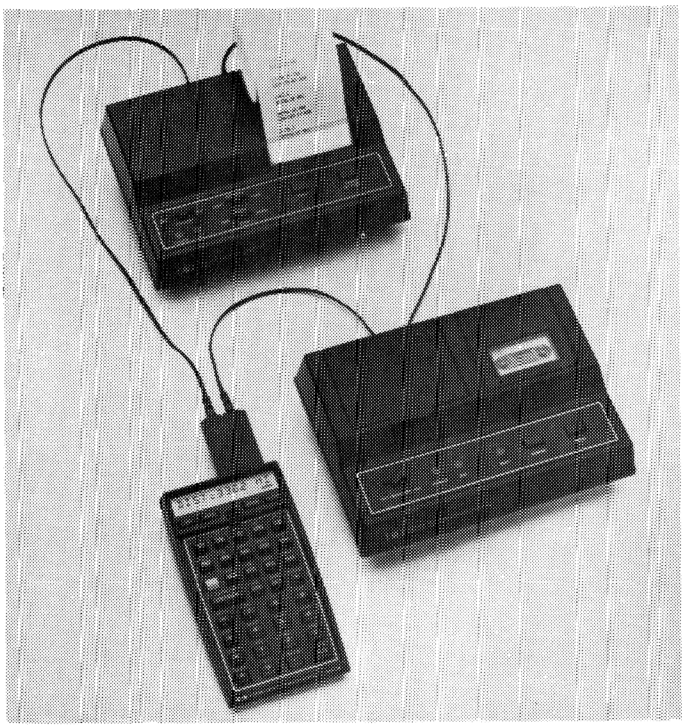
The 24K bytes of ROM in the HP-41CX features an RPN operating system. This sytem permits users to view intermediate results as well as recover easily from error conditions. The RAM in the HP-41CX retains its contents when the unit is turned off. An alphanumeric LCD screen permits viewing of both numeric and alphabetical characters plus special symbols and operators. The alphanumeric keyboard is redefinable. Thus, users may assign frequently used programs or functions to a single key to customize operation. Keyboard overlays are available for users who want to create their own labeled keyboards.

The HP-41CX can be expanded, via the HP-IL (Hewlett-Packard Interface Loop) . Peripherals include printers, plotters and instruments. All of these can be maintained as a battery-operated system allowing complete portability.

For more information on the $325.00 HP-41CX, contact your local HP sales office.

Photo *HP-41CX Programmable Calculator.*

Photo *HP Peripherals Connected Via HP-IL.*



ENHANCED FILE MAINTENANCE PROGRAM

One of the satisfying aspects of being the Editor of *PCN* is seeing the improvements that come about over time as a consequence of publishing hard to obtain information. As you know, during the past year (and more), *PCN* has published a great deal of information on machine language programming for the Sharp PC-1500 and Radio Shack PC-2. We are now seeing the results of having made this type of information widely available.

The program presented here is an excellent example of the types of improvements that can be obtained through the judicious application of ML techniques. This is an enhanced version of the File Maintenance program that originally appeared in Issue 24 of *PCN*. It was developed by the originator of the program, *Stephen Tomback, 19 Maplewood Way, Pleasantville, NY 10570*. Through the use of machine language routines, Mr. Tomback has vastly improved the overall operating speed of the program. Additionally, he has removed the former barrier (imposed by BASIC) that limited record lengths to 80 characters. If you found the original version useful, you will be ecstatic over this one!

Overview

The File Maintenance program will allow the set up of a filing system to a user's specific needs. Think of a file box filled with 3 x 5 cards. The box is called a *file*, the cards are called *records*, and the lines on the cards are called *fields*.

This programs allows the user to define the fields on the cards, store information in those fields, and have the ability to retrieve cards by searching fields for specific information. Thus, one could utilize this file maintenance program as a filing system for such things as birthdays, anniversaries, magazine subscriptions, or investments. A periodic search (such as monthly) would serve as a reminder as to which items were coming due.

Equipment Required

This program operates in a Radio Shack PC-2 or a Sharp PC-1500 that is equipped with an 8K memory expansion module. In order to make use of the program you will also need a printer/cassette interface such as the Sharp CE-150.

About the File

Here is important information for you to know about the structure of the file:

Number of Records: The program utilizes just under 3K of memory. The remainder is used for storage of the file. Using an 8K memory expansion module gives the PC a total memory capacity of about 10 kilobytes. Thus, about 7 kilobytes would be available for data storage. The number of records that can be stored will depend on the total number of bytes assigned to a record. For instance, a file utilizing 100 bytes per record would allow for the storing of approximately 70 records.

File Name: When a file is created, it must be assigned a name. This name may not exceed 15 characters. The assigned name is saved with the file. (Using a file name that is shorter than 15 bytes will not increase the amount of data that can be stored.)

Field Name: A record may contain up to 12 fields. (A field is simply an item within a record. A name, an address, and a telephone number might each constitute a field.) When a file is created, the user must assign a name to each field. That name must not exceed 15 characters. Again, assigning less than 15 characters to a field name will not make additional room for the storage of data.

Field Length: The length of a field may be set at from 1 to 40 bytes.

Record Length: The record length is a function of the number of field(s) defined and their length(s). Since a field may have a maximum of 40 bytes and a record may contain up to 12 fields, the maximum record length is 480 bytes. (A record this size would not be practical in most situations because it would only allow approximately 14 records in a file.)

Fields Per Record: A record may have from 1 to 12 fields.

Field Search: A record may be searched by 1 to 4 fields. If you set up a file of birthdays as well as anniversaries, it would be beneficial to establish a field called TYPE. This field would be used to distinguish between the two categories (say B and A, respectively). Another field might be named MONTH DUE. Finding all the birthdays that occurred in a particular month could then be accomplished by a two field search.

Before Starting

The first time the RUN command is issued, the program POKES several machine language routines into memory. It then erases those POKE commands from the BASIC program in order to save memory. Attempting to copy the program after it has been started should thus be avoided.

Also, using the RUN command will cause any file currently in memory to be erased. Do *not* use RUN to restart the program once you have any part of a file stored in memory. Always use the directive GOTO 15 to restart the program without loss of data. (You might want to place such a statement under control of a softkey.)

The program uses the black pen of the printer. Make sure you have a good pen in that position.

When selecting from the menu, remember that

Program *File Maintenance*

```
10:GOSUB 995:LOCK
  :TEXT :CSIZE 1
  :COLOR 0
15:CLS :WAIT 0:D=
  9:H=0:RESTORE
  :FOR I=1TO D:
  READ I$(0):
  PRINT I$(0)
20:FOR J=1TO 30:I
  $=INKEY$ :IF I
  $<>""LET J=60
25:NEXT J
30:IF I$=""THEN 5
  0
35:FOR J=1TO D:IF
  MID$ ("NOI1AVP
  FC",J,1)=I$LET
  H=J:J=D
40:NEXT J
45:IF HCLS :ON H
  GOSUB 100,200,
  300,400,500,70
  0,700,800,900:
  GOTO 15
50:NEXT I:GOTO 15
55:DATA "N - NEW
  FILE","O - OUT
  PUT/TAPE","I -
   INPUT/TAPE"
60:DATA "1 - FILE
   PARAMETERS","
  A - ADD RECORD
  ","P - PRINT R
  ECORD"
65:DATA "V - VIEW
   RECORD","F -
  FIND RECORD(S)
  ","C - CHANGE
  RECORD"
100:IF LEN N$=0
  THEN 115
105:BEEP 3:INPUT "
  SURE? ";I$
110:IF I$<>"Y"
  RETURN
115:GOSUB 995
120:INPUT "NEW FIL
  E NAME: ";N$
125:IF LEN N$<1OR
  LEN N$>15THEN
  120
130:INPUT "FIELDS
  PER RECORD: ";
  K
135:IF K<1OR K>12
  THEN 130
140:FOR I=1TO K
145:CLS :PRINT "FI
  ELD";I;" NAME:
  ":CURSOR 14:
  INPUT "";@$(I+
  14)
150:IF LEN @$(I+14
  )<1OR LEN @$(I
  +14)>15THEN 14
  5
155:CLS :INPUT "HO
  W MANY BYTES:
  ";@(I+14)
160:IF @(I+14)<1OR
  @(I+14)>40THEN
  155
165:L=L+@(I+14):
  NEXT I
170:GOSUB 190
175:PRINT "MAXIMUM
    RECORDS";N:
  BEEP 15:RETURN
190:N=INT ((STATUS
  3-STATUS 2-63)
  /L):RETURN
200:IF LEN N$=0
  GOSUB 960:
  RETURN
205:INPUT "TAPE RE
  ADY? ";I$
210:IF I$<>"Y"
  RETURN
215:PRINT #N$;@$(*
  ),@(*)
220:I$=N$+"*":
  CSAVE MI$;
  STATUS 2,
  STATUS 2+(M*L)
  :RETURN
300:IF LEN N$=0
  THEN 315
305:BEEP 3:INPUT "
  SURE? ";I$
310:IF I$<>"Y"
  RETURN
315:GOSUB 995:
  INPUT "FILE NA
  ME: ";N$
320:INPUT #N$;@$(*
  ),@(*)
325:GOSUB 190
330:IF N<MPRINT "F
  ILE SPACE EXCE
  EDED!":BEEP 15
  :GOSUB 995:
  RETURN
335:I$=N$+"*":
  CLOAD MI$;
  STATUS 2:
  RETURN
400:IF LEN N$=0
  GOSUB 960:
  RETURN
405:LF 5:LPRINT "F
  ILENAME: ";N$
410:LPRINT "MAXIMU
  M RECORDS:";N
415:LPRINT "RECORD
  S USED:";M
420:LPRINT "BYTES
  PER RECORD:";L
  :LF 1
425:FOR I=1TO K:
  LPRINT "FIELD"
  ;I;" ";@$(I+14
  );TAB 28;USING
  "###";"BYTES";
  @(I+14):USING
430:NEXT I:LF 6:
  RETURN
500:IF LEN N$=0
  GOSUB 960:
  RETURN
505:IF M=NPRINT "O
  UT OF RECORDS!
  ":BEEP 15:
  RETURN
510:F=STATUS 2+(M*
  L):FOR I=1TO K
  :G=I+14
515:CLS :I$(0)="":
  PRINT @$(G);"
  ";CURSOR LEN @
  $(G)+2:INPUT "
  ";I$(0)
520:IF LEN I$(0)>@
  (G)GOSUB 985:
  GOTO 515
525:GOSUB 955:F=F+
  @(G):NEXT I:
  CLS :M=M+1:
  PRINT "ADD REC
  ORD";M:BEEP 15
  :RETURN
600:LF 1:LPRINT "R
  ECORD:";I:LF 1
605:G=STATUS 2+((I
  -1)*L):FOR F=1
  TO K:E=F+14
610:CALL STATUS 3-
  5,@(E):CALL
  STATUS 3-14,G:
  J=STATUS 3+7:
  CALL STATUS 3-
  35,J:G=G+@(E)
615:IF I$<>"V"
  LPRINT STR$ (F
  );" ";I$(0):
  NEXT F:RETURN
620:J=1
625:PRINT STR$ (F)
  ;" ";MID$ (I$(
  0),J,24)
630:IF ASC INKEY$
  =8LET J=J+1:
  GOTO 650
635:IF ASC INKEY$
  =12LET J=J-1:
  GOTO 650
640:IF ASC INKEY$
  <>13THEN 630
645:CLS :NEXT F:
  RETURN
650:IF J=0LET J=1
655:IF J>LEN I$(0)
  LET J=LEN I$(0
  )
660:GOTO 625
700:IF M=0GOSUB 96
  0:RETURN
705:I=0:INPUT "REC
  ORD NUMBER. ";
  I
710:IF I<1OR I>M
  THEN RETURN
715:IF I$="V"GOSUB
  605:RETURN
720:GOSUB 600:LF 5
  :RETURN
800:IF M=0GOSUB 96
  0:RETURN
805:CLS :H=0:INPUT
  "HOW MANY FIEL
  D SEARCH? ";H
810:IF H<1OR H>4OR
  H>KTHEN RETURN
815:RESTORE 885:
  FOR I=1TO H:
  READ I$:J$=
  STR$ I
820:CLS :@(I)=0:
  CURSOR :PRINT
  J$;I$:CURSOR 4
  :INPUT "FIELD
  NUMBER: ";@(I)
825:IF @(I)<1OR @(
  I)>KTHEN 820
830:CLS :I$(0)="":
  CURSOR :PRINT
  "SEARCH";@(I);
  " FOR: ":
  CURSOR 14:
```

```
    INPUT "",I$(0)
835:IF LEN I$(0)>1
    6OR LEN I$(0)>
    @(@(I)+14)
    GOSUB 985:GOTO
    830
840:IF LEN I$(0)<1
    THEN 830
845:@$(I)=I$(0):
    NEXT I
850:FOR I=1TO M:
    FOR J=1TO H
855:CLS :PRINT "RE
    CORD";I
860:G=@(J):GOSUB 9
    90
865:I$="":FOR E=1
    TO LEN @$(J):I
    $=I$+CHR$ PEEK
    F:F=F+1:NEXT E
870:IF MID$ (I$,1,
    E-1)<>@$(J)
    NEXT I:RETURN
875:NEXT J
880:GOSUB 600:NEXT
    I:RETURN
885:DATA "st","nd"
    ,"rd","th"
900:IF M=0GOSUB 96
    0:RETURN
905:I=0:INPUT "REC
    ORD NUMBER: ",
    I
910:IF I<1OR I>M
    THEN RETURN
915:G=0:INPUT "FIE
    LD NUMBER: ";G
920:IF G<1OR G>K
    THEN 915
925:I$(0)="":INPUT
    "CHANGE TO: ";
    I$(0)
930:IF LEN I$(0)>@
    (G+14)GOSUB 98
    5:GOTO 925
935:GOSUB 990:G=G+
    14:GOSUB 955:
    RETURN
955:CALL STATUS 3-
    5,@(G):CALL
    STATUS 3-14,F:
    J=STATUS 3+7:
    CALL STATUS 3-
    63,J:RETURN
960:PRINT "FILE EM
    PTY!":BEEP 15:
    RETURN
985:CLS :PRINT "TO
    O LONG!":BEEP
    15:RETURN
990:F=STATUS 2+((I
    -1)*L)
992:IF G=1RETURN
994:FOR E=1TO G-1:
    F=F+@(E+14):
    NEXT E:RETURN
995:CLEAR :DIM I$(
    0)*41
996:POKE STATUS 3-
    35,&A5,&76,&D0
    ,&DF,&2A,&A5,&
    76,&D1,&18,&A5
    ,&76,&D2,&1A,&
    55,&41,&88
997:POKE STATUS 3-
    19,4,&B5,0,&41
    ,&9A,&84,&AE,&
    76,&D1,4,&AE,&
    76,&D2,&9A,4,&
    AE,&76,&D0,&9A
998:POKE STATUS 3-
    63,&A5,&76,&D0
    ,&DF,&2A,&A5,&
    76,&D1,&18,&A5
    ,&76,&D2,&1A,&
    45
999:POKE STATUS 3-
    49,&B7,0,&8B,4
    ,&51,&88,8,&9A
    ,&B5,32,&51,&8
    8,3,&9A
1000:POKE STATUS
    2,&84,&28,4,
    &2A,&BE,&D2,
    &EA,0,&A5,&7
    8,&A6,8,&A5
1005:POKE STATUS
    2+13,&78,&A7
    ,&A,&46,&46,
    &46,&FB,&A4,
    &AE,&76,&D0,
    &9A
1010:POKE STATUS
    2+25,&84,&AE
    ,&78,&67,4,&
    AE,&78,&68,&
    9A
1015:I=996.CALL
    STATUS 2,I:
    IF PEEK &76D
    0=&BSTOP
1020:X=STATUS 2-I
    -27
1025:A=STATUS 2-1
    -X
1030:CALL STATUS
    2+25,A
1035:POKE I-1,&3A
    ,&F1,&99,&D,
    3,&E4,&14,&F
    1,&AB,&53,&2
    0,&54,&4F
1040:POKE I+12,&4
    D,&42,&41,&4
    3,&4B,&20,&4
    6,&4D,&20,&3
    8,&2F,&38,&3
    3,&D,&FF
1045:POKE I-15,&1
    1
1050:CLEAR :DIM I
    $(0)*41:
    RETURN
STATUS 1      3566
```

only a single keystroke is needed. All other entries, such as data, must be terminated by pressing the ENTER key. For instance, to establish a new file just depress the N key when the menu is cycling. To enter a new file name, type in the desired name followed by the ENTER key.

At the Beginning

Load the program into the computer, place the PC into the RUN mode and type RUN. A continuously cycling menu will be displayed containing the following items:

```
N  -  NEW FILE
O  -  OUTPUT/TAPE
I  -  INPUT/TAPE
1  -  FILE PARAMETERS
A  -  ADD RECORD
V  -  VIEW RECORD
P  -  PRINT RECORD
F  -  FIND RECORD(S)
C  -  CHANGE RECORD
```

An Illustration

To show how the program may be applied, a file for holding birthdays and anniversaries will be created. It will be named AUTOCALENDAR. The seven fields will be titled and sized as follows:

	Name	Length (Bytes)
Field 1:	TYPE	1
Field 2:	MONTH DUE	2
Field 3:	NAME	20
Field 4:	ADDRESS	20
Field 5:	CTY ST ZP	23
Field 6:	PHONE	12
Field 7:	NOTE	16

Setting Up A New File

A new file is created using the NEW FILE option. Depress the N key while the menu is cycling and NEW FILE NAME will be displayed. If a file already exists the programs asks SURE? Any response other than Y at this point will return the program to the main menu. Responding Y clears the memory of any previous file. The program will then prompt for

a NEW FILE NAME. This procedure permits a user who has inadvertently selected the NEW FILE function to exit without erasing the existing file.

If you desire to follow along with this example, enter the name AUTOCALENDAR for the new file name. The program will then ask FIELDS PER RECORD. Enter the number 7 for this illustration. (Remember, the maximum permitted is 12, a value greater than that will not be accepted.)

The PC will then request the name of each field and its length in characters. Respond as outlined previously if you want to follow the example. After the file has been set up, the program will issue a beep, then display the maximum number of records that may be stored (74 in this example). After this the program will go back to displaying the menu.

Printing File Parameters

This option causes the current filename, maximum number of records that can be stored in the file, the number of records used, the number of bytes in a record, and each field name and its length to be printed. After outputting this information the program reverts to displaying the menu.

If you are following the example procedure, it is a good idea to select this option and verify that you have set up the file correctly. Depress the number 1 digit key while the menu is cycling to select this function. If you find that the file is not set up as you desire, then stop the program with the BREAK key. You would then issue a RUN command to restart the program.

It is important to realize that once defined, you cannot alter the file parameters without losing all the data currently in memory. Make sure you have things set up properly before you start building up a file of valuable data!

Adding A Record

Now that a file has been created and its proper format verified, try adding a birthday to it. Depress the A key while the menu is cycling. The first field description (TYPE:) will appear. Respond B to this field for birthday. (Remember, field 1 in this application is used used to differentiate between anniversaries [A] and birthdays [B].)

The second field description, MONTH DUE, will then be displayed. Respond with a two-digit answer. Thus, 01 should be used for January. The value 12 would be used for December. (If a single digit was used for January, then a single character search of the MONTH DUE field for the digit 1 would find both the 1 and the 12 entries. Using two characters means that a properly specified search will yield exactly the desired match.)

Continue entering the rest of the information for the record as prompted by the field names. (You might want to use the NOTE field for the actual birthdate or some other relevant information.)

When all the fields in the record have been inputted, the computer will beep. It will then show the record number assigned to that entry before going back to display the menu.

Viewing A Record

To display the contents of a record on the LCD, use this option. It is selected by pressing the V key when in the menu mode. Respond to the prompt for RECORD NUMBER by entering the desired value. (To view the example record just entered, respond with a 1, assuming it is the first entry in your file.)

The number that appears on the lefthand side of the screen while viewing is the field number. This is followed by the name of the field and then its contents. Press ENTER to display each consecutive field within a record.

The LCD can display 24 characters of the fields numbered 1 through 9 and 23 characters of the fields numbered 10 through 12. However, the forward and back arrow keys may be used to scroll the screen right and left when fields exceed these lengths.

When the last field in a record has been displayed, pressing the ENTER key will return the program to the cycling menu.

Printing A Record

Use the PRINT RECORD option, selected by pressing the P key, to list a record on the printer. (To list the example record, respond to the prompt for RECORD NUMBER with a 1.) When the record has been printed, control returns to the menu.

Changing A Record

Depress the C key when in the menu mode to bring up this option. It permits the altering of data within a specified field in a designated record. Respond appropriately to the prompts for RECORD NUMBER and FIELD NUMBER. When the prompt CHANGE TO is displayed, enter the desired data.

If the new information is different in length than the original field entry, the remainder of the field is filled with spaces. To delete the contents of a field, respond to the CHANGE TO prompt by pressing just the ENTER key. This causes the entire field to be filled with spaces.

After a field has been changed, the program returns to the menu mode.

Finding Record(s)

Records in a file may be located using this option. From one to four fields may be searched at a time. After you have inputted several birthdays and anniversaries into the example file, try finding a

particular birthday by month and name as follows:

Depress the F key while in the menu mode to bring up the prompt HOW MANY FIELD SEARCH? For purposes of illustration, assume a three-field search on the fields named TYPE, MONTH DUE and NAME.

Now respond to the 1st FIELD NUMBER prompt with a 1 to indicate that the first field (TYPE) is to be checked. Respond to SEARCH 1 FOR with a B for birthday in this example.

Respond to the 2nd FIELD NUMBER prompt with a 2 as the second field (MONTH DUE) is to be examined. Respond to the prompt SEARCH 2 FOR with the month of the birthday you are seeking.

Respond to 3rd FIELD NUMBER with a 3 to indicate that the third field (NAME) is to be inspected. Respond to SEARCH 3 FOR with the name of the person you are seeking.

After this entry the program will begin looking for the record that matches all the parameters requested. As it searches it will display the record number that it is examining. When it finds a record matching all the specifications, it will output it to the printer. When the search has been completed, the program returns to the menu mode.

It should be noted that a field search may be performed in any order. While field numbers 1, 2 and 3 were specified in the example, they could have been requested in the order 3, 1, and then 2.

Saving A File On Tape

Files may be saved on tape. Make sure the tape recorder is properly connected and set to the record mode. Check that the cassette has been advanced beyond the tape leader. Press the O key to bring up the TAPE READY? query. Any response other than Y will terminate the procedure. A response of Y will cause the file to be recorded on tape using the name of the current file. (For instance, AUTOCALENDAR, in the case of the current example.) When the file has been written, program operation reverts to the menu.

Since data files can *not* be verified, it is a good idea to make a second copy of important files on a second tape cassette. Just load in another tape and repeat the tape save process.

Restoring A File From Tape

Files that have been previously stored on tape may, of course, be read back for use by this program. Make sure that the tape unit is properly connected and in the playback mode. Press the I key to bring up the prompt SURE? An answer of Y at this point clears the memory of any previous file. The program will then prompt for the FILE NAME. This procedure provides an exit from inadvertent selection of the INPUT/TAPE option without erasing an existing file.

Inputting the name of a file (such as the example AUTOCALENDAR) causes the program to begin reading the tape. The file names it finds are displayed on the LCD.

A data file is actually saved in two parts by this program. Hence, when it is recovered, it will be processed as one file under the original user-assigned name (AUTOCALENDAR for example) and another file (AUTOCALENDAR*) that has the same name suffixed by an asterisk. The program will automatically process both files. When the procedure has been completed, the display will return to the cycling menu.

Deleting Records

In order to save memory, a delete option was not included in the program. However, if a record that has been created is no longer needed, then alter one of its fields to signify that it has been deleted. For instance, place the letter D in field 1. When you need a new record, use the FIND RECORD(S) option to locate such a deleted record. Then use the CHANGE RECORD option to edit in your new data.

RABBIT CHASE

David Hergert, 4714 Chickering Ave., Cincinnati, OH 45232 submitted this clever and challenging game. It is designed to run on the Sharp PC-1500 or Radio Shack PC-2 equipped with at least 4K of RAM. It uses the printer to draw graphics at the beginning of the program. Additionally, if you successfully complete a game by concluding the fourth level, another set of graphics is drawn as a reward.

The game itself is played on the LCD. In the game, the player is pictured as a rabbit that is trying to eat all the vegetables in Farmer Brown's garden. Farmer Brown is trying to catch the rabbit with his tractor. The objective is for the rabbit to eat all of the vegetables on all four levels of play, before Farmer Brown catches it. Each level of play is harder than the previous one. (Level four is really tough!)

Pressing the number 4 key causes the rabbit to move to the right. The number 6 key changes its direction to the left. The rabbit has wrap-around capabilities, but the tractor does not. Don't let the tractor catch the rabbit!

If you become totally frustrated and unable to win the game at level four, you can see the final graphics by entering RUN 950. If you want to skip the opening printer graphics, use RUN 20. The normal start is to simply issue a RUN command.

Listing *Rabbit Chase*

```
  5:FOR X=25TO 40:
    BEEP 1,X,X*2:
    NEXT X
 10:TEXT :COLOR 0
 15:LPRINT TAB 3;"
    RABBIT CHASE":
    LF 4:GOSUB 700
 20:POKE 20509,72,
    118,74,0,5,189
    ,68,64,65,78,7
    8,153,9
 30:POKE 20522,76,
    119,139,6,72,1
    19,74,0,158,18
    ,154
 40:WAIT 0:Z=1:DIM
    Y(5):Y(1)=75:Y
    (2)=120:Y(3)=7
    5:Y(4)=120
 50:O=1:N=11:X=0:Y
    =Y(Z):A2=147:A
    1=10:IF Z=5
    THEN 950
 60:IF Z>2LET N=12
 80:CLS
 90:CALL 20509:
    POKE &7701,&44
 95:GCURSOR X:
    GPRINT "000000
    00002070707C30
    "
100:IF INKEY$ ="4"
    GCURSOR X:
    GPRINT "307C70
    70200000000000
    ":X=X-4:IF X<=
    0THEN LET X=14
    7:GOTO 300
110:IF INKEY$ ="4"
    AND A1>=Y+10
    AND A2>=Y
    GCURSOR Y:
    GPRINT "3C783B
    3F780000000000
    0000":Y=Y-2:
    GOTO 100
120:IF INKEY$ ="4"
    AND A1<=YAND A
    2<=Y+10GCURSOR
    Y:GPRINT "3C78
    3B3F7800000000
    00":Y=Y-2:GOTO
    100
130:IF A1>Y-1AND A
    2>YGCURSOR Y:
    GPRINT "3C783B
    3F780000000000
    00":GOTO 160
140:IF A1<YAND A2<
    Y+11GCURSOR Y:
    GPRINT "3C783B
    3F780000000000
    ":GOTO 160
150:GCURSOR Y:
    GPRINT "3C783B
    7F384040404040
    "
160:Y=Y-2
170:IF A1>=A2-NLET
    Z=Z+1:GOTO 50
180:IF Y<=X+4
    LPRINT "GOTCHA
    ":CLS :INPUT "
    NEW GAME?(Y/N)
     ";BB$:IF BB$=
    "Y"CLEAR :GOTO
    40
190:IF BB$="N"END
200:IF INKEY$ ="4"
    AND Y<=XLPRINT
    "GOTCHA":CLS :
    INPUT "NEW GAM
    E?(Y/N) ";BB$:
    IF BB$="Y"
    CLEAR :GOTO 40
210:X=X+2:IF X>A1
    LET A1=X
220:IF INKEY$ ="4"
    THEN 100
230:GOTO 95
300:GCURSOR 0:
    GPRINT 0;0;0;0
    ;0;0;0;0;0;0;0
310:IF X<=A2-10LET
    A2=X
320:GCURSOR X:
    GPRINT "307C70
    70200000000000
    "
330:IF INKEY$ ="6"
    GCURSOR X:
    GPRINT "000000
    00002070707C30
    ":X=X+4:IF X<1
    47THEN 370
340:IF INKEY$ ="6"
    AND X>146LET X
    =0:GCURSOR 147
    :GPRINT 0;0;0;
    0;0;0;0;0;0;0:
    GOTO 95
350:IF A2>101LET O
    =1
360:IF A2<=101AND
    Z>2LET O=3
370:IF A1<=YAND A2
    <=YGCURSOR Y-2
    :GPRINT "00000
    0000000783F3B7
    83C":GOTO 400
380:IF A1>=Y-9AND
    A2>=YGCURSOR Y
    :GPRINT "00000
    00000783F3B783
    C":GOTO 400
390:GCURSOR Y:
    GPRINT "404040
    4040783F3B783C
    "
400:IF A1>=A2-NLET
    Z=Z+1:GOTO 50
410:Y=Y+O
420:IF Y>=X-4
    LPRINT "GOTCHA
    ":CLS :INPUT "
    NEW GAME?(Y/N)
     ";BB$:IF BB$=
    "Y"CLEAR :GOTO
    40
430:IF BB$="N"END
435:IF INKEY$ ="6"
    THEN 330
440:X=X-2:IF X<A2
    LET A2=X
450:GOTO 300
700:GRAPH :LINE (0
    ,0)-(155,40),2
    ,3,B
710:LINE (0,0)-(15
    5,-40),2,3,B
720:LINE (0,-40)-(
    155,-80),2,3,B
730:LINE (20,-80)-
    (15,-80)-(15,-
    78)-(13,-78)-(
    13,-72),0,2
731:LINE (13,-72)-
    (17,-68)-(19,-
    68)-(22,-70)
740:LINE (22,-70)-
    (22,-65)-(24,-
    63)-(25,-62)-(
    23,-70)-(26,-7
    4)
741:LINE (26,-74)-
    (23,-76)-(22,-
    76)-(20,-78)-(
    15,-78)
750:LINE (22,-76)-
    (20,-78)-(15,-
    78)
800:LINE (20,-40)-
    (15,-40)-(15,-
    38)-(13,-38)-(
    13,-32)
801:LINE (13,-32)-
    (17,-28)-(19,-
    28)-(22,-30)
805:LINE (22,-30)-
    (22,-25)-(24,-
    23)-(25,-22)-(
    23,-30)-(26,-3
    4)
806:LINE (26,-34)-
    (23,-36)-(22,-
    36)-(20,-38)-(
    15,-38)
810:LINE (20,0)-(1
    5,0)-(15,2)-(1
    3,2)-(13,8)
811:LINE (13,8)-(1
    7,12)-(19,12)-
    (22,10)
815:LINE (22,10)-(
    22,15)-(24,17)
    -(25,18)-(23,1
    0)-(26,6)
816:LINE (26,6)-(2
    3,4)-(22,4)-(2
    0,2)-(15,2)
820:LINE (20,40)-(
    15,40)-(15,42)
    -(13,42)-(13,4
    8)
821:LINE (13,48)-(
    17,52)-(19,52)
    -(22,50)
825:LINE (22,50)-(
    22,55)-(24,57)
    -(25,58)-(23,5
    0)-(26,46)
826:LINE (26,46)-(
    23,44)-(22,44)
    -(20,42)-(15,4
    2)
830:LINE (0,40)-(0
    ,-80),0,1
835:LINE (155,40)-
    (155,-80)
840:LINE (100,45)-
    (100,62)-(114,
    62)-(114,55)-(
    132,55)-(132,4
    5)
841:LINE (132,45)-
    (100,45):LINE
    (105,40)-(110,
    45),,,B:LINE (
    120,40)-(125,4
    5),,,B
842:LINE (120,72)-
    (125,67),,,B:
```

```
    LINE (125,70)-
    (125,55)
850:LINE (50,5)-(5
    0,22)-(64,22)-
    (64,15)-(82,15
    )-(82,5)
851:LINE (82,5)-(5
    0,5):LINE (55,
    0)-(60,5),,,B:
    LINE (70,0)-(7
    5,5),,,B
852:LINE (70,32)-(
    75,27),,,B:
    LINE (75,30)-(
    75,15)
860:LINE (100,-35)
    -(100,-18)-(11
    4,-18)-(114,-2
    5)-(132,-25)-(
    132,-35)
861:LINE (132,-35)
    -(100,-35):
    LINE (105,-40)
    -(110,-35),,,B
    :LINE (120,-40
    )-(125,-35),,,
    B
862:LINE (120,-8)-
    (125,-13),,,B:
    LINE (125,-10)
    -(125,-25)
870:LINE (50,-75)-
    (50,-58)-(64,-
    58)-(64,-65)-(
    82,-65)-(82,-7
    5)
871:LINE (82,-75)-
    (50,-75):LINE
    (55,-80)-(60,-
    75),,,B:LINE (
    70,-80)-(75,-7
    5),,,B
872:LINE (70,-48)-
    (75,-53),,,B:
    LINE (75,-50)-
    (75,-65)
900:COLOR 1
910:TEXT :LF -6:
    TAB 13:LPRINT
    "LEVEL"
915:TAB 14:LPRINT
    "4"
920:LF 1:TAB 14:
    LPRINT "3"
925:LF 1:TAB 14:
    LPRINT "2"
930:LF 1:TAB 14:
    LPRINT "1"
935:LF 2:LPRINT "P
    RESS 4 TO MO
    VE LEFT":
    LPRINT "PRESS
     6 TO MOVE RI
    GHT":LF 2
940:IF INKEY$ ="S"
    LPRINT "YOU AR
    E ON":RETURN
945:WAIT 0:PRINT "
    PRESS S TO S
    TART":GOTO 940
950:FOR X=40TO 1
    STEP -1:BEEP 1
    ,X,X*2:NEXT X
960:TEXT :LF 4:
    GRAPH
970:LINE (150,0)-(
    160,10)-(180,1
    0)-(185,20)-(2
    00,20),0,3
971:LINE (200,20)-
    (190,0)-(150,0
    ):LINE (155,0)
    -(160,-5),,,B
972:LINE (175,0)-(
    180,-5),,,B:
    LINE (167,10)-
    (180,30),0,2:
    LINE (180,30)-
    (190,40),,,B
973:LINE (173,17)-
    (183,17)
975:LINE (20,0)-(2
    4,35)-(35,46)-
    (40,45)-(55,42
    )-(60,42),0,1
976:LINE (60,42)-(
    70,42)-(75,62)
    -(69,70)-(62,6
    0)-(60,42)-(70
    ,42)
977:LINE (70,42)-(
    80,35)-(78,20)
    -(69,15)-(52,0
    )-(20,0)
978:LINE (50,-10)-
    (25,-10)-(25,0
    ):LINE (66,31)
    -(69,34),,,B:
    LINE (24,33)-(
    18,39),,,B
980:COLOR 0
990:TEXT :LF 5:
    LPRINT "SUPER
    RABBIT!":END
STATUS 1       3847
```

ALTERNATE CONTROL OF THE PC-1500 LCD

This is the third and concluding part of this series. The challenge I put forth at the end of the previous article (in Issue 29 of *PCN*) was for you to work on converting the LINE routine from BASIC to ML. That routine was presented in Issue 28 of *PCN* and had been assigned BASIC line numbers 5000 - 5140. It is used to plot the pixels that must be turned on in order to draw a smooth straight line between two pairs of X,Y coordinates on the LCD screen. Converting the LINE routine to machine language will greatly increase the speed at which lines can be drawn.

There are, of course, almost an infinite number of ways in which one could approach such a project. Machine language offers such a wide range of instructions and approaches that virtually no two programmers would produce the exact same series of code for such a project. Thus, the sequence of instructions you might have used to approach this problem could be completely different than those that I chose.

A Parallel Course

In order to provide a comparative approach for readers, I elected to essentially duplicate the BASIC routine using ML methods. In particular, I chose to manipulate the same BASIC variables (M, N, R, S, U and V). I used this method primarily so that those of you who may not have a lot of ML experience could observe how a ML routine can closely pattern a BASIC routine. This technique does not provide for the most efficient or fastest running ML program. However, even this relatively crude approach offers vastly increased speed over that of BASIC. For, while the same variables are being manipulated, the "interpretive" process that slows down BASIC has been removed.

The ML listing on the accompanying pages shows the ML code that replaces lines 5000 through 5060 of the original BASIC program (refer to *PCN* Issue 28). This represents roughly half of the LINE routine. In order to conserve space (since the commented ML listing is rather lengthy), the second half of the converted LINE routine (representing lines 5070 - 5140) is provided simply as an object code listing. It is similar in format to the first half. (Those with a deep interest can use a disassembler to study this part of the code in detail.)

The entire machine code for the LINE routine fits within exactly 512 bytes of memory. I assigned this code to the address range &0000 - &01FF since I have a 16K RAM module in the PC I used for development of this package. However, the entire code in this 512-byte block is relocatable. Thus, if

Listing *First Half of LINE Routine Converted To Machine Language.*

PG	LC	B1	B2	B3	B4	B5	LABELS	MNEMONICS	COMMENTS
00	00	58	7A				LINE	LDYH #7A	Value required by ROM FP routine.
00	02	48	79					LDXH #79	Establish pointer to BASIC variable S.
00	04	4A	90					LDXL #90	Establish pointer to BASIC variable S.
00	06	BE	F7	3F				JSR F73F	ROM subroutine to transfer S into FP register R0.
00	09	E6						CALL E6	ROM subroutine to transfer FP register R0 into R2.
00	0A	48	79					LDXH #79	Establish pointer to BASIC variable R.
00	0C	4A	88					LDXL #88	Establish pointer to BASIC variable R.
00	0E	BE	F7	3F				JSR F73F	ROM subroutine to transfer R into FP register R0.
00	11	BE	EF	B6				JSR EFB6	ROM subroutine to subtract FP registers R0-R2 (R-S).
00	14	A5	7A	02				LDA 7A02	Fetch first byte of R0 mantissa into the accumulator.
00	17	8B	E7					FBZ 5070	Jump to work 2nd dimension when S=R (mantissa is zero).
00	19	A5	7A	01				LDA 7A01	Fetch sign byte of R0 into the accumulator.
00	1C	8B	06					FBZ S<R	If sign byte is zero, then S less than R.
00	1E	68	00					LDUH #00	When sign byte negative, S>R, so set register U=60001.
00	20	6A	01					LDUL #01	Finish setting register U=60001.
00	22	8E	04					FB UINR0	Jump to place contents of register U into FP register R0.
00	24	68	FF				S<R	LDUH #FF	When sign byte positive, S<R, so set register U=-1.
00	26	6A	FF					LDUL #FF	Finish setting register U=-1 (6FFFF).
00	28	CD	10				UINR0	CALL 10	ROM subroutine to place CPU register U into R0.
00	2A	80						80	Byte that needs to be passed for ROM subroutine.
00	2B	48	7A					LDXH #7A	Value required by ROM FP routine.
00	2D	58	79					LDYH #79	Establish pointer to BASIC variable P.
00	2F	5A	78					LDYL #78	Establish pointer to BASIC variable P.
00	31	BE	F7	11				JSR F711	ROM subroutine to transfer R0 into BASIC variable P.
00	34	48	79					LDXH #79	Establish pointer to BASIC variable R.
00	36	4A	88					LDXL #88	Establish pointer to BASIC variable R.
00	38	58	79					LDYH #79	Establish pointer to BASIC variable M.
00	3A	5A	60					LDYL #60	Establish pointer to BASIC variable M.
00	3C	BE	F7	33				JSR F733	ROM subroutine to transfer BASIC variable R into M.
00	3F	38						NOP	Save a byte for alterations??? (Safety valve.)
00	40	58	7A				5040	LDYH #7A	Value required by ROM FP routine.
00	42	48	79					LDXH #79	Establish pointer to BASIC variable U.
00	44	4A	A0					LDXL #A0	Establish pointer to BASIC variable U.
00	46	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable U to R0.
00	49	E6						CALL E6	Rom subroutine to transfer FP register R0 to R2.
00	4A	48	79					LDXH #79	Establish pointer to BASIC variable V.
00	4C	4A	A8					LDXL #A8	Establish pointer to BASIC variable V.
00	4E	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable V to R0.
00	51	BE	EF	B6				JSR EFB6	ROM subroutine to subtract R2 from R0 (V-U).
00	54	BE	DB	F5				JSR DBF5	ROM subroutine to push R0 (V-U) onto FP stack.
00	57	48	79					LDXH #79	Establish pointer to BASIC variable R.
00	59	4A	88					LDXL #88	Establish pointer to BASIC variable R.
00	5B	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable R to R0.
00	5E	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	5F	48	79					LDXH #79	Establish pointer to BASIC variable S.
00	61	4A	90					LDXL #90	Establish pointer to BASIC variable S.
00	63	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable S to R0.
00	66	BE	EF	B6				JSR EFB6	ROM subroutine to subtract R2 from R0 (S-R).
00	69	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	6A	BE	DC	16				JSR DC16	ROM subroutine to pop FP stack into R0 (V-U).
00	6D	CD	58					CALL 58	ROM subroutine to divide R0 by R2 ((V-U)/(S-R)).
00	6F	BE	DB	F5				JSR DBF5	ROM subroutine to push R0 ((V-U)/(S-R)) onto FP stack.
00	72	48	79					LDXH #79	Establish pointer to BASIC variable R.
00	74	4A	88					LDXL #88	Establish pointer to BASIC variable R.
00	76	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable R to R0.
00	79	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	7A	48	79					LDXH #79	Establish pointer to BASIC variable M.
00	7C	4A	60					LDXL #60	Establish pointer to BASIC variable M.
00	7E	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable M to R0.
00	81	BE	EF	B6				JSR EFB6	ROM subroutine to subtract R2 from R0 (M-R).
00	84	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.

PG	LC	B1	B2	B3	B4	B5	LABELS	MNEMONICS	COMMENTS
00	85	BE	DC	16				JSR DC16	ROM subroutine to pop FP stack into R0 ((V-U)/((S-R)).
00	88	CD	7E					CALL 7E	ROM subroutine to multiply R0 by R2 ((V-U)/(S-R))*(M-R).
00	8A	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	8B	48	79					LDXH #79	Establish pointer to BASIC variable T. Note: T=0.5!
00	8D	4A	98					LDXL #98	Establish pointer to BASIC variable T.
00	8F	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable T to R0.
00	92	F0						CALL F0	ROM subroutine to add R2 to R0 ((V-U)/(S-R))*(M-R)+T.
00	93	BE	F5	BE				JSR F5BE	ROM subroutine to form integer of quantity in R0.
00	96	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	97	48	79					LDXH #79	Establish pointer to BASIC variable U.
00	99	4A	A0					LDXL #A0	Establish pointer to BASIC variable U.
00	9B	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable U to R0.
00	9E	F0						CALL F0	Add R2 to R0 to form (INT((V-U)/(S-R))*(M-R)+0.5)+U.
00	9F	D0						CALL D0	ROM subroutine to convert R0 to CPU register U.
00	A0	04	00					04 00	Bytes that need to be passed for ROM subroutine.
00	A2	FD	A8					PSHU	Save contents of register U on processor stack.
00	A4	58	7A					LDYH #7A	Value required by ROM FP routine.
00	A6	48	79					LDXH #79	Establish pointer to BASIC variable M.
00	A8	4A	60					LDXL #60	Establish pointer to BASIC variable M.
00	AA	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable M to R0.
00	AD	D0						CALL D0	ROM subroutine to convert R0 to CPU register U.
00	AE	04	00					A0 00	Bytes that need to be passed for ROM subroutine.
00	B0	FD	28					LDX U	Load CPU register X with contents of CPU register U.
00	B2	BE	70	50				JSR 7050	Pass X-coordinate value to X-coordinate display routine.
00	B5	FD	0A					POPX	Pop former contents of CPU register U into CPU register X.
00	B7	BE	70	A0				JSR 70A0	Pass Y-coordinate value to Y-coordinate display routine.
00	BA	38						NOP	Save a byte for alterations??? (Safety valve.)
00	BB	38						NOP	Save a byte for alterations??? (Safety valve.)
00	BC	38						NOP	Save a byte for alterations??? (Safety valve.)
00	BD	58	7A					LDYH #7A	Value required by ROM FP routine.
00	BF	48	79					LDXH #79	Establish pointer to BASIC variable P.
00	C1	4A	78					LDXL #78	Establish pointer to BASIC variable P.
00	C3	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable P to R0.
00	C6	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	C7	48	79					LDXH #79	Establish pointer to BASIC variable M.
00	C9	4A	60					LDXL #60	Establish pointer to BASIC variable M.
00	CB	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable M to R0.
00	CE	F0						CALL F0	ADD M to P to get new value of M (loop variable).
00	CF	BE	DB	F5				JSR DBF5	ROM subroutine to push R0 (M) onto FP stack.
00	D2	48	7A					LDXH #7A	Value required by ROM FP routine.
00	D4	58	79					LDYH #79	Establish pointer to BASIC variable M.
00	D6	5A	60					LDYL #60	Establish pointer to BASIC variable M.
00	D8	BE	F7	11				JSR F711	ROM subroutine to transfer R0 to BASIC variable M.
00	DB	A5	79	79				LDA 7979	Fetch sign byte of BASIC variable P into accumulator.
00	DE	8B	0E					FBZ SUBM	If P is positive, go calculate S-M.
00	E0	58	7A					LDYH #7A	Value required by ROM FP routine. (Prepare for M-S.)
00	E2	48	79					LDXH #79	Establish pointer to BASIC variable S.
00	E4	4A	90					LDXL #90	Establish pointer to BASIC variable S.
00	E6	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable S to R0.
00	E9	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	EA	CD	30					CALL 30	ROM subroutine to pop M from FP stack to FP register R0.
00	EC	8E	0A					FB SUBTCT	P is negative, go calculate M-S.
00	EE	CD	30				SUBM	CALL 30	ROM subroutine to pop variable M from FP stack into R0.
00	F0	E6						CALL E6	ROM subroutine to transfer FP register R0 to R2.
00	F1	48	79					LDXH #79	Establish pointer to BASIC variable S.
00	F3	4A	90					LDXL #90	Establish pointer to BASIC variable S.
00	F5	BE	F7	3F				JSR F73F	ROM subroutine to transfer BASIC variable S to R0.
00	F8	BE	EF	B6			SUBTCT	JSR EFB6	Subtract R2 from R0 (S-M when P+, M-S when P-).
00	FB	A5	7A	01				LDA 7A01	Fetch sign byte of R0 into the accumulator.
00	FE	9B	C0					RBZ 5040	Loop back to process next display position on positive R0.
01	00	58	7A				5070	LDYH #7A	Begin second part of the LINE subroutine. . . .

Listing *Second Half of LINE Routine.*

```
0100 58 7A 48 79   0180 3F BE EF B6
0104 4A A8 BE F7   0184 E6 BE DC 16
0108 3F E6 48 79   0188 CD 58 E6 48
010C 4A A0 BE F7   018C 79 4A 98 BE
0110 3F BE EF B6   0190 F7 3F F0 BE
0114 A5 7A 02 8B   0194 F5 BE E6 48
0118 E6 A5 7A 01   0198 79 4A 88 BE
011C 8B 06 68 00   019C F7 3F F0 D0
0120 6A 01 8E 04   01A0 04 00 FD 28
0124 68 FF 6A FF   01A4 BE 70 50 58
0128 CD 10 80 48   01A8 7A 48 79 4A
012C 7A 58 79 5A   01AC 68 BE F7 3F
0130 70 BE F7 11   01B0 D0 04 00 FD
0134 48 79 4A A0   01B4 28 BE 70 A0
0138 58 79 5A 68   01B8 38 38 38 38
013C BE F7 33 38   01BC 58 7A 48 79
0140 58 7A 48 79   01C0 4A 70 BE F7
0144 4A A0 BE F7   01C4 3F E6 48 79
0148 3F E6 48 79   01C8 4A 68 BE F7
014C 4A 68 BE F7   01CC 3F F0 BE DB
0150 3F BE EF B6   01D0 F5 48 7A 58
0154 BE DB F5 48   01D4 79 5A 68 BE
0158 79 4A 88 BE   01D8 F7 11 A5 79
015C F7 3F E6 48   01DC 71 8B 0E 58
0160 79 4A 90 BE   01E0 7A 48 79 4A
0164 F7 3F BE EF   01E4 A8 BE F7 3F
0168 B6 E6 BE DC   01E8 E6 CD 30 8E
016C 16 CD 7E BE   01EC 0A CD 30 E6
0170 DB F5 48 79   01F0 48 79 4A A8
0174 4A A0 BE F7   01F4 BE F7 3F BE
0178 3F E6 48 79   01F8 EF B6 A5 7A
017C 4A A8 BE F7   01FC 01 9B BF 9A
```

you are using, say, a CE-155 (8K) module, then you could place this block of code in the range &3800 - &39FF. (Just remember to use a NEW &XXXX command before loading the code to protect the ML area. On my 16K system I use NEW &0200. If you had a CE-155 you would use NEW &4000. And, if you relocate the code, then you must be sure and change the address of the BASIC CALL statement that is used to access this code!)

Before attempting to use the code presented here, you must also install the two ML routines provided in the previous issue (29) of *PCN*. These start at addresses &7050 and &70A0. This is vital as the LINE routine calls upon these subroutines. (They are called by jump to subroutine instructions in the ML code for the LINE routine.)

The BASIC Connection

With the ML routines installed in memory (which replace the original BASIC LINE and ON routines), all one needs to do to draw a line on the LCD is to do the following: Define the starting coordinates of the line by setting the variables R,U (for X- and Y-coordinate positions). Define the ending points by setting the variables S,V. Call the new LINE routine in this manner:

```
5000:"LINE"T=.5:CALL &0000:RETURN
```

(Note that the BASIC variable T must be defined as the constant 0.5 prior to calling the machine code. This value is used by the ML routines. While it could have been set by the ML routines, doing so would have caused them to consume more than 512 bytes.)

Want to see your PC quickly draw a box on the LCD? Just use the following BASIC statements that call on the ML line-drawing routines:

```
1000 CLS
1010 R=0:U=0:S=6:V=0:GOSUB "LINE"
1020 R=6:U=0:S=6:V=6:GOSUB "LINE"
1030 R=6:U=6:S=0:V=6:GOSUB "LINE"
1040 R=0:U=6:S=0:V=0:GOSUB "LINE"
1050 END
```

A Few Words of Caution

ML programming gives you greater control over the operation of the PC. However, using it requires greater responsibility of the programmer. For instance, if the values of R, S, V and U are not within valid ranges (0-155 for R and S, 0-6 for U and V), then the ML routines can end up stuffing garbage into RAM locations, ultimately leading to a "crash" of the PC.

Thus, it was with grief that I found out that I had made an error in the BASIC line doodler routine that I provided in Issue 28 of *PCN*. Line 310 of that routine should read as follows:

```
310 R=RND (156)-1:
    S=RND (156)-1:
    U=RND (7)-1:V=
    RND (7)-1
```

The original statements in that line could lead to out of range values being generated (where the X-coordinate exceeded 155 or Y exceeded 6). Make sure you correct this line if you want to use that random line generator routine to test the ML routines!

Now You Are One Your Own

You now have the capability of dealing with the LCD screen as an X,Y plane. You can rapidly draw lines between selected coordinates. As illustrated earlier, you can connect lines to form boxes. Of course, this concept can be extended to create elaborate drawings. You have a new tool in your software library. How you use and apply this new capability is up to you.

FROM THE WATCH POCKET

Here it is, the end of our third year of publication. As is the custom in the year-end edition, I will take a little extra space for this column. This enables me to review progress in the past year as well as speculate on what may occur in the year ahead.

Not Much Happened

The past year was certainly not a stellar one in terms of PC progress. Hardly anything that might be considered revolutionary was introduced on the hardware side. Most of what might be called real progress boils down to just a few minor model changes.

Perhaps the most important advance came when Sharp was convinced that 2K of RAM was not enough for today's sophisticated PC users. The Sharp PC-1500A is an exact replica of the PC-1500 (right down to the ROM chips) with some more internal RAM. In the new PC-1500A there is 6K of user RAM in the address range &4000 through &57FF. There is another 1K of RAM, apparently intended for ML routines, tucked in the address range &7C00 through &7FFF. Just don't use the first byte of that area as it occassionally gets stuffed with a &0D when the BASIC input buffer overflows. (A software quirk?) Add in the area dedicated to storage of string and numeric variables and you have the 8K of memory Sharp is entitled to advertise. For all practical purposes, however, your BASIC programs must fit in 6K of memory, not the 8K you might assume.

Despite the hopes and prayers of many users, there was very little in the way of peripherals added by the PC manufacturers. Most notably absent was the introduction of any kind of practical external mass storage device. Of course, Sharp may view their 16K RAM module, backed up by batteries, as a viable alternative. Personally, I do not. At about $150 per clip, it is a long way from what I consider low cost mass storage. Of course, you can always spend a half-hour loading 16K of RAM from tape cassette, provided you don't catch a glitch.

Hurray For The Little Guys

Perhaps the most exciting news of the year is being generated by a small firm in Pennsylvania. *Robert Undi* of **Usonics**, *7901 Oak Hill Drive, Cheltenham, PA 19012*, reports that they have successfully mated a "stringy floppy" data drive to the PC-1500. Priced at about $400.00, this unit gives users the capability of storing and recovering data at a rate that allows 16K to be transferred in about 30 seconds. Supplies of the special drive are said to be somewhat limited. But, if you are looking for something about 50 times faster than audio cassette, you might look into this item.

Usonics also sells the new Brother EP-22 portable typewriter. This is the advanced version of the earlier EP-20. The beauty of this model is that it has a built-in RS-232C interface. Just connect it to your CE-158 RS-232C interface and you have a highly portable external printer. The EP-22 is priced at about $250.00. When not being used as a printer, it can serve as an electronic portable typewriter, complete with 2K of internal, editable, memory! A big cut above the earlier model, and at just a $50.00 premium, this seems like a good buy.

Put Your Program(s) On ROMs?

A growing trend among PC users in businesses is to develop packages of programs that can be used by many employees. Groups of these programs are then "burned" into ROMs for general distribution.

Several firms are now engaged in providing ROM programming services. Out in Indiana, a firm by the name **Carleton Micro Systems** is providing a complete modular package that can hold 8K or 16K of EPROM. The module is intended primarily for businesses, institutions or software vendors that need 25 or more packages of the same program(s). The firm takes software supplied on cassette or a battery-backed RAM module and converts it to their EPROM units. The modules disable the editing, program listing and store functions of the PC-1500 so that the installed software cannot be copied. In lots of 25, the 16K modules sell for approximately $150.00 each plus a software setup fee.

If you don't need 25 or more copies of a program and you can get by with just 8K of program storage, you might be interested in the module duplication service offered by **Atlantic N.E. Marketing**, *P.O. Box 921, Marblehead, MA 01945*. Duplication of your own 8K (battery-backed) module starts at about $100.00 for a single unit and declines as the quantity increases.

OEM Action Increasing

The pocket computer market currently seems to be flat to declining at the consumer level, but increasing in the OEM market. In other words, the PC is being repackaged by others to end up with a product that performs a specific function or serves a particular purpose.

Thus, there is now a company that sells a PC-1500 dressed up as a Bond Analyzer. For about $700.00 you can get a system, completely pre-programmed, that calculates just about everything a businessperson would want to know about bond prices, yields, etc. Another firm makes a specialty of providing PC-1500s to banks, all set up to

1983 Index -- Ordered By Author

Last Name	First Name	Title	Issue
Auersbacher	Emerich	Wind Chill Equivalent Temp.	21
Auersbacher	Emerich	Count-Down Timer Program	27
Bean	Norman	Renumbering Utility	29
Beckman	Mel	Sidelister Program	22
Campbell	Diane	Cross Reference for the PC-2	27
Carter	Don	26K of RAM in the PC-1500/PC-2	23
Cloutier	David	Star Targets	29
Delinger	Willian	Controlling Lamp Using CE-158	28
Delinger	Willian	Hexadecimal Loader	28
Doughty	Russell	Histogram on PC-2/PC-1500	21
Etheridge	Steven	Fractional Base Conversion	28
Fieux	Robert	Using the CE-158 Interface	28
Frey	Michael	RPN Calculator for Programmers	26
Hergert	David	Meteors Game for PC-2/PC-1500	28
Hergert	David	Rabbit Chase	30
Jackson	David	Expanded Display Routine	23
Motto	David	Epson HX-20 Review	21
Motto	David	Epson HX-20 Follow-Up	23
Norton	John	Graphing on PC-2/PC-1500	24
Norton	John	BASIC Line Deletion	24
Organ	R.	Fuel Consumption Analyst	29
PCN	Staff	Radio Shack Announces the PC-3	27
PCN	Staff	Sharp Steps Up In Size	26
PCN	Staff	The Sharp CE-125 Printer/Microcassette Interface	23
PCN	Staff	Radio Shack Offers PC-4	23
PCN	Staff	The Sharp PC-1251	22
PCN	Staff	Epson HX-20 Weather Forecaster	22
PCN	Staff	Texas Instrument's CC-40	22
PCN	Staff	Sharp Introduces Wallet-Size PC	21
PCN	Staff	Radio Shack Micro Color Computer	25
PCN	Staff	TRS-80 Model 100 Sets Trend	25
PCN	Staff	The CE-158 Interface	24
PCN	Staff	PC-1250/51 Memory Dump Program	24
PCN	Staff	Radio Shack TRS-80 Model 100	24
PCN	Staff	Cassette Volume Monitor	27
PCN	Staff	Paper Holder Kit for PC-1500 & PC-2	27
PCN	Staff	Memory Maps	23
PCN	Staff	Casio Software Modules	28
PCN	Staff	Casio FX-802P Includes Printer	28
PCN	Staff	Gavilan Readying Mobile Computer	28
PCN	Staff	Alternate Control of the PC-1500 LCD	28
PCN	Staff	Hewlett-Packard HP-41CX	30
PCN	Staff	Carlton 16K EPROM Modules	30
PCN	Staff	Alternate Control of the PC-1500 LCD	29
PCN	Staff	Casio FX-700P Product Review	29
PCN	Staff	RoadRunner Portable Computer	29
PCN	Staff	Alternate Control of the PC-1500 LCD	30
Pollett	Patrick	Redefining the PC-1500 Keyboard	29
Rober	Norlin	The CE-502A General Statistics Module	26
Rober	Norlin	Inside the Sharp PC-1250	26
Rober	Norlin	Machine-Language Monitor for the PC-1500/PC-2	22
Rober	Norlin	Normal Probability Using Machine Language	25
Rober	Norlin	PC-1500/PC-2 Cassette Storage	25
Rober	Norlin	Understanding PC-1500 Part 4	21
Rober	Norlin	Three-Dimensional Plotting	21
Rober	Norlin	Three-Dimensional Projections	27
Rober	Norlin	Vertical Lister	27
Stock	Robert	RS-232C Interface Print Formatter	28
Thomas	Arthur	Date & Time on the PC-2/PC-1500	28
Tomback	Stephen	File Maintenance Program	24
Tomback	Stephen	Enhanced File Maintenance	30

1983 Index -- Ordered By Article

Title	Last Name	First Name	Issue
26K of RAM in the PC-1500/PC-2	Carter	Don	23
Alternate Control of the PC-1500 LCD	PCN	Staff	28
Alternate Control of the PC-1500 LCD	PCN	Staff	29
Alternate Control of the PC-1500 LCD	PCN	Staff	30
BASIC Line Deletion	Norton	John	24
Carlton 16K EPROM Modules	PCN	Staff	30
Casio FX-700P Product Review	PCN	Staff	29
Casio FX-802P Includes Printer	PCN	Staff	28
Casio Software Modules	PCN	Staff	28
Cassette Volume Monitor	PCN	Staff	27
Controlling Lamp Using CE-158	Delinger	Willian	28
Count-Down Timer Program	Auersbacher	Emerich	27
Cross Reference for the PC-2	Campbell	Diane	27
Date & Time on the PC-2/PC-1500	Thomas	Arthur	28
Enhanced File Maintenance	Tomback	Stephen	30
Epson HX-20 Follow-Up	Motto	David	23
Epson HX-20 Review	Motto	David	21
Epson HX-20 Weather Forecaster	PCN	Staff	22
Expanded Display Routine	Jackson	David	23
File Maintenance Program	Tomback	Stephen	24
Fractional Base Conversion	Etheridge	Steven	28
Fuel Consumption Analyst	Organ	R.	29
Gavilan Readying Mobile Computer	PCN	Staff	28
Graphing on PC-2/PC-1500	Norton	John	24
Hewlett-Packard HP-41CX	PCN	Staff	30
Hexadecimal Loader	Delinger	Willian	28
Histogram on PC-2/PC-1500	Doughty	Russell	21
Inside the Sharp PC-1250	Rober	Norlin	26
Machine-Language Monitor for the PC-1500/PC-2	Rober	Norlin	22
Memory Maps	PCN	Staff	23
Meteors Game for PC-2/PC-1500	Hergert	David	28
Normal Probability Using Machine Language	Rober	Norlin	25
Paper Holder Kit for PC-1500 & PC-2	PCN	Staff	27
PC-1250/51 Memory Dump Program	PCN	Staff	24
PC-1500/PC-2 Cassette Storage	Rober	Norlin	25
Rabbit Chase	Hergert	David	30
Radio Shack Announces the PC-3	PCN	Staff	27
Radio Shack Micro Color Computer	PCN	Staff	25
Radio Shack Offers PC-4	PCN	Staff	23
Radio Shack TRS-80 Model 100	PCN	Staff	24
Redefining the PC-1500 Keyboard	Pollett	Patrick	29
Renumbering Utility	Bean	Norman	29
RoadRunner Portable Computer	PCN	Staff	29
RPN Calculator for Programmers	Frey	Michael	26
RS-232C Interface Print Formatter	Stock	Robert	28
Sharp Introduces Wallet-Size PC	PCN	Staff	21
Sharp Steps Up In Size	PCN	Staff	26
Sidelister Program	Beckman	Mel	22
Star Targets	Cloutier	David	29
Texas Instrument's CC-40	PCN	Staff	22
The CE-158 Interface	PCN	Staff	24
The CE-502A General Statistics Module	Rober	Norlin	26
The Sharp CE-125 Printer/Microcassette Interface	PCN	Staff	23
The Sharp PC-1251	PCN	Staff	22
Three-Dimensional Plotting	Rober	Norlin	21
Three-Dimensional Projections	Rober	Norlin	27
TRS-80 Model 100 Sets Trend	PCN	Staff	25
Understanding PC-1500 Part 4	Rober	Norlin	21
Using the CE-158 Interface	Fieux	Robert	28
Vertical Lister	Rober	Norlin	27
Wind Chill Equivalent Temp.	Auersbacher	Emerich	21

determine mortgage rates or calculate consumer loans. Still another organization supplies heating contractors with a PC-1500 system programmed to solve heating and air conditioning problems, as well as deliver a sales pitch to prospective customers. The sales pitch generates charts and graphs illustrating how much money a new energy-efficient heating/cooling system can save over its estimated lifespan. I have even heard about an ambitious project that will help airline pilots keep on course as they fly all over the country. A PC-1500, programmed with all kinds of technical data, can calculate the best flight plans, advise on mid-flight course corrections, and serve as a general navigation aid. There are people working on the same type of capability for mariners, too.

Many of these systems will appear on the market packaged so that the actual users will hardly even know that what they are really using is just a specially programmed PC. In other words, the PC is becoming an OEM instrument. It is being used as the vehicle in which special-purpose vendors simply package their product. After all, their real product is simply the software that gives the PC the ability to perform a particular type of task.

The Trend Is Definitely Towards Bigger!

There is not doubt that people want portability. But, pocketable units are having a hard time keeping up with the expectations and demands of the public. People are willing to pay for full size keyboards and larger displays. The appetite for more memory is probably insatiable. Witness the success of notebook-sized units such as the Radio Shack Model 100 and Epson HX-20. There will be lots more new entries of this approximate size during the coming year.

Why, even Sharp may forget what a PC is in its rush to push its new PC-5000. (By the way, I hear the unit is now finding its way into stores in the United States. Check with your local Sharp dealer if you have been anxiously waiting to examine an actual unit. Price of a standard 128K unit is reportedly $1995.00. Add about $400.00 if you want the optional integrated printer.)

Looking For A/D?

A firm by the name of **Enertec**, *558 Highland Ave., Upper Montclair, NJ 07043*, has developed an interface for the PC-1500 & PC-2 that includes: 8 digital inputs, 8 digital outputs, and an A to D converter with 8 analog inputs. The company developed the interface for a special project. However, it is considering manufacturing the unit for general distribution. Write to *Paul Schwartz* at the company address or phone (201) 744-4962 and speak to him if you would like to see such a product on the market.

PCN To Stay With PCs.

During 1984 we plan to continue devoting the bulk of our space to the most popular "pocketable" units, primarily the Sharp PC-1500/1500A and Radio Shack PC-2. Of course, we will follow new entries and keep an eye on the smaller notebook portables.

If you have been wondering what happened to our popular feature PC-1500 writer and internationally respected PC whiz **Norlin Rober**, never fear, he has been hard at work compiling a massive report on the PC-1500's ROM. We plan to provide an ongoing comprehensive series revealing all its secrets during the coming year, thanks to Norlin's monumental efforts.

There are lots of other goodies planned too, such as the powerful Little EDitor (LED) program that will be featured in the next issue. This is really a line oriented word processor program that is sure to thrill and amaze many PC-1500/PC-2 owners.

We Have Come A Long Way

Current editions of *PCN* are now four times larger than when we first started publishing. We are most definitely rather large for a "newsletter." I wish to extend my personal thanks to all our faithful subscribers who have helped make this growth possible. Best wishes for a healthy and prosperous New Year to everyone!

Available Only by Prepaid Subscription for a Calendar Year Period (January - December). You are sent back issues for the calendar year to which you subscribe, at the time you enroll.

MC/VISA Phone subscriptions: (203) 888-1643

__ I am interested. Please send more information. I have:
__ a Sharp PC-1500 __ Radio Shack PC-2 __ ?_____

__ Enroll me as a 1984 Subscriber (Issue numbers 31-36). $18.00 in U.S. ($21.00 to Canada/Mexico. Elsewhere U.S. $30.00 payable in U.S. funds against a U.S. bank.)

__ Enroll me as a 1983-84 Subscriber (Issue numbers 21-36). $54.00 in U.S. ($63.00 to Canada/Mexico. Elsewhere U.S. $80.00 payable in U.S. funds against a U.S. bank.)

__ Enroll me as a 1982-84 Subscriber (Issue numbers 11-36). $78.00 in U.S. ($95.00 to Canada/Mexico. Elsewhere U.S. $120.00 payable in U.S. funds against a U.S. bank.)

__ Check here if paying by MasterCard or VISA. Please give credit card information below.

Orders must be accompanied by payment in full.

Name: _____

Addr: _____

City: _____ State: ____ Zip: _____

MC/VISA #: _____

Signature: _____ Exp. Date: _____

POCKET COMPUTER NEWSLETTER
P.O. Box 232, Seymour, CT 06483